Information-Theoretic Mechanics of Survival in Maximum-Entropy Environments

Static Minimax Disturbance Attenuation, Coordination-Deficit Costs, and the Placed Source as Categorical Right Adjoint

Formal Proof Companion

Charles D. Schaper, Ph.D.
Annals of the Chrysene Formalism, Volume 1, Number 4
Article 97, November 2026

1 Scope and kernel audit

The supplied formal listing was consulted and replaced. The final Lean 4.24.0 source freshly compiles **43 named theorems**. Forty have no axiom dependencies. Three use the standard logical axiom `propext`: `projection_additive`, `projection_surjective`, and `plaquette_xor`. No theorem depends on `sorryAx`, native-evaluation axioms or user-declared assumptions.

These declarations include an actual support-poset adjunction with arbitrary natural-number species labels and arbitrary supports containing a species-dependent target on an arbitrary registry type, rather than merely a finite table of category counts. The remaining certificates cover finite algebra, graph expansion and arithmetic. They do not formalize the Sturm spectral argument, logarithmic/thermal analysis, continuous norm optimization, thermodynamic work or molecular realization.

2 Audit of the supplied listing

The supplied `kekule_sachs_split` and `adjunction_triangles` proofs fail under Lean 4.24 because `rfl` does not construct the displayed conjunctions. Their subsequent printed dependencies include `sorryAx`. The additive projection and one functional composition identity compile, but the old functions do not encode the manuscript's support categories. In particular, the listed arithmetic $8^2 = 64$ is not a proof of a graph's matching count or determinant. The final source replaces this with explicit graph matrices and signed/unsigned recursive row expansion.

3 Correspondence to the revised manuscript

Support-poset category.
An expanded object carries a species, a Boolean support on a parameterized registry type, and proof that its species-dependent target belongs to it. A morphism requires equal species and reverse support inclusion. The full support defines F; species projection defines G. Twelve declarations prove identities, composition, functor actions, hom equivalence and inverse maps, the identity unit, counit, triangle identities, and postcomposition naturality. Thinness supplies equality of parallel arrows. These are general statements for this explicit category, not an abstract claim about a physical preparation process. A finite registry is recovered by restricting the ambient support to a finite universe.

Projection and graph calculations.
Exhaustive binary projection certificates check additivity, surjectivity and fibres; a concrete multiplication counterexample prevents reading this as a field homomorphism. The two 9×9 bipartite blocks are explicit. Recursive unsigned row expansion gives 8 and 5 perfect matchings; signed expansion gives determinants 8 and 5 for the blocks. The separate $-64/-25$ declarations verify the arithmetic implied by the analytic block-determinant identity, not a formal library theorem identifying an 18×18 determinant.

Other finite consequences.
Certificates cover bounded zero-slack binomial values, species/support cardinalities, the illustrative strain values, complete-site valency, plaquette parity and count, Boolean feasibility, and capacity arithmetic. The capacity slope and radius constants are finite integer inequalities used in the companion's alternative analytic all-length proof. The actual irrational tile-floor enclosure is checked in Python, not Lean.

The registry type and its species-to-target function are parameters of the category. Taking the registry type to be a finite binary-word type gives the manuscript model directly. Species labels use natural numbers; restricting to the finite set of chemical words preserves the same thin-category arguments. No claim about entropy or physical realizability follows from the category alone.

4 Reproduction

Use Lean 4.24.0, then run `LEAN=/path/to/lean python audit_lean.py`. The script verifies the source hash, recompiles, and checks the exact dependency declaration for every named theorem. The log, source pin, JSON audit and toolchain file accompany the package. Both PDF and standalone LaTeX embed the exact audited source below.

5 Complete reproducible source

Article 97 source

```
/- Article 97 revised finite certificates, Lean 4.24 core only.
Actual support-poset adjunction is modeled; analytic and chemical claims
are outside this file. -/
namespace Article97
set_option maxRecDepth 100000
set_option maxHeartbeats 0

def bit (x i : Nat) : Nat := x / 2^i % 2
def proj (x : Nat) : Nat := x % 4

def Bchr : List (List Int) := [[1, 0, 0, 0, 0, 0, 1, 0, 0], [1, 1, 0, 0, 0, 0, 0, 0, 0],
     [0, 0, 1, 0, 0, 0, 0, 0, 1], [0, 0, 0, 1, 1, 0, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0, 1,
    0], [0, 0, 0, 0, 0, 1, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0, 1, 1], [0, 1, 0, 0, 0, 0, 1,
    0, 1], [0, 0, 1, 1, 0, 0, 0, 1, 0]]
def Btet : List (List Int) := [[1, 0, 0, 0, 0, 1, 0, 0, 0], [0, 1, 0, 0, 0, 1, 1, 0, 0],
     [0, 0, 1, 0, 0, 0, 1, 1, 0], [0, 0, 0, 1, 0, 0, 0, 1, 1], [0, 0, 0, 0, 1, 0, 0, 0,
    1], [1, 1, 0, 0, 0, 0, 0, 0, 0], [0, 1, 1, 0, 0, 0, 0, 0, 0], [0, 0, 1, 1, 0, 0, 0,
    0, 0], [0, 0, 0, 1, 1, 0, 0, 0, 0]]
def entry (b : List (List Int)) (r c : Nat) : Int := (b[r]?.getD [])[c]?.getD 0
-- Signed/unsigned row expansion over remaining columns.
def expand (b : List (List Int)) (signed : Bool) : Nat -> List Nat -> Int
 | 0, _ => 1
 | n+1, cols => ((cols.zipIdx).map (fun (c,j) =>
     if entry b (8-n) c == 0 then 0 else
     entry b (8-n) c * (if signed then (-1)^j else 1) *
     expand b signed n (cols.erase c))).sum
-- Actual pointed support-poset, arbitrary species and supports.
structure Expanded (R : Type) (t : Nat -> R) where
 species : Nat
 support : R -> Bool
 target_mem : support (t species) = true

variable {R : Type} {t : Nat -> R}

def Hom (y z : Expanded R t) : Prop :=
 y.species = z.species /\ (forall r, z.support r = true -> y.support r = true)
def F (R : Type) (t : Nat -> R) (s : Nat) : Expanded R t := { species := s, support :=
    fun _ => true, target_mem := rfl }
def G (y : Expanded R t) : Nat := y.species
def homTo (s : Nat) (y : Expanded R t) (h : Hom (F R t s) y) : s = G y := h.1
def homFrom (s : Nat) (y : Expanded R t) (h : s = G y) : Hom (F R t s) y :=
 And.intro h (fun _ _ => rfl)

theorem hom_identity (y : Expanded R t) : Hom y y := by
 exact And.intro rfl (fun _ h => h)
theorem hom_composition (x y z : Expanded R t) (h : Hom x y) (k : Hom y z) : Hom x z :=
    by
 exact And.intro (Eq.trans h.1 k.1) (fun r hr => h.2 r (k.2 r hr))
theorem G_functor (x y : Expanded R t) (h : Hom x y) : G x = G y := h.1
theorem F_functor (R : Type) (t : Nat -> R) (s u : Nat) (h : s = u) : Hom (F R t s) (F R
     t u) := by
 cases h
 exact hom_identity (F R t s)
theorem adjunction_hom (s : Nat) (y : Expanded R t) : Hom (F R t s) y <-> s = G y := by
 exact Iff.intro (homTo s y) (homFrom s y)
theorem hom_bijection_left (s : Nat) (y : Expanded R t) (h : Hom (F R t s) y) :
 homFrom s y (homTo s y h) = h := by rfl
theorem hom_bijection_right (s : Nat) (y : Expanded R t) (h : s = G y) :
 homTo s y (homFrom s y h) = h := by rfl
theorem unit_identity (R : Type) (t : Nat -> R) (s : Nat) : G (F R t s) = s := by rfl
theorem counit_exists (y : Expanded R t) : Hom (F R t (G y)) y := homFrom (G y) y rfl
theorem triangle_F (R : Type) (t : Nat -> R) (s : Nat) : counit_exists (F R t s) =
    hom_identity (F R t s) := by rfl
```

```
theorem triangle_G (y : Expanded R t) : G_functor (F R t (G y)) y (counit_exists y) = (
    rfl : G y = G y) := by rfl
theorem hom_naturality (s : Nat) (y z : Expanded R t) (h : Hom y z) (k : Hom (F R t s) y
    ) :
 homTo s z (hom_composition (F R t s) y z k h) = Eq.trans (homTo s y k) (G_functor y z h
    ) := by rfl

theorem projection_additive : forall u v : Fin 8,
 proj (Nat.xor u.val v.val) = Nat.xor (proj u.val) (proj v.val) := by decide
theorem projection_surjective : forall y : Fin 4, exists x : Fin 8, proj x.val = y.val
    := by
 intro y
 exact Exists.intro (Fin.mk y.val (Nat.lt_trans y.isLt (by decide))) (Nat.mod_eq_of_lt y
    .isLt)
theorem projection_kernel : ((List.range 8).filter (fun x => proj x == 0)).length = 2 :=
     by decide
theorem projection_fibres : forall y : Fin 4, ((List.range 8).filter (fun x => proj x ==
     y.val)).length = 2 := by decide
-- alpha^2 projects to0 in GF8 basis; in GF4 alpha^2=alpha+1=3.
theorem projection_not_multiplicative : proj 4 != 3 := by decide

theorem chrysene_permanent : expand Bchr false 9 (List.range 9) = 8 := by decide
theorem tetracene_permanent : expand Btet false 9 (List.range 9) = 5 := by decide
theorem chrysene_signed_expansion : expand Bchr true 9 (List.range 9) = 8 := by decide
theorem tetracene_signed_expansion : expand Btet true 9 (List.range 9) = 5 := by decide
theorem chrysene_edge_count : (Bchr.map List.sum).sum = 21 := by decide
theorem tetracene_edge_count : (Btet.map List.sum).sum = 21 := by decide
theorem chrysene_adjacency_det_value : -(8^2 : Int) = -64 := by decide
theorem tetracene_adjacency_det_value : -(5^2 : Int) = -25 := by decide

def choose : Nat -> Nat -> Nat
 | _, 0 => 1
 | 0, _+1 => 0
 | n+1, k+1 => choose n k + choose n (k+1)
theorem zero_slack : forall n : Fin 10, choose n.val n.val = 1 := by decide
theorem registry_count : 2^5 = 32 := by decide
theorem species_count : 3^6 = 729 := by decide
theorem support_count : 2^31 = 2147483648 := by decide
theorem slack_registry : choose 32 1 = 32 := by decide
theorem strain_three : 6*3^3 = 162 := by decide
theorem strain_two : 4*3^2 = 36 := by decide
theorem strain_difference : 162-36 = 126 := by decide
theorem six_neighbors : 6-(4+2) = 0 := by decide
theorem plaquette_xor : forall a b c d : Fin 4,
 Nat.xor (Nat.xor a.val b.val) (Nat.xor (Nat.xor b.val c.val) (Nat.xor (Nat.xor c.val d.
    val) (Nat.xor d.val a.val))) = 0 := by decide
theorem box_faces : 2*2*6 + 2*3*5 + 3*2*5 = 84 := by decide
theorem feasibility : forall a b c : Bool, (a && b && c) = true <-> a = true /\ b = true
     /\ c = true := by decide
theorem interior_bits : 2*233+4*54 = 682 := by decide
theorem boundary_bits : 2*2377 = 4754 := by decide
theorem capacity_example : 682 < 4754 := by decide
-- Rational all-L sufficient-bound constants; analytic implication in prose.
theorem radius_bound_constant : 26923^2*233 > 16*100000^2 := by decide
theorem capacity_slope_constant : 24*63^2 > 100^2 := by decide
theorem capacity_margin_constant : 2*(24*63^2-100^2)*233 > 218*100^2 := by decide
end Article97

#print axioms Article97.hom_identity
#print axioms Article97.hom_composition
#print axioms Article97.G_functor
#print axioms Article97.F_functor
#print axioms Article97.adjunction_hom
#print axioms Article97.hom_bijection_left
#print axioms Article97.hom_bijection_right
#print axioms Article97.unit_identity
#print axioms Article97.counit_exists
#print axioms Article97.triangle_F
```

```
#print axioms Article97.triangle_G
#print axioms Article97.hom_naturality
#print axioms Article97.projection_additive
#print axioms Article97.projection_surjective
#print axioms Article97.projection_kernel
#print axioms Article97.projection_fibres
#print axioms Article97.projection_not_multiplicative
#print axioms Article97.chrysene_permanent
#print axioms Article97.tetracene_permanent
#print axioms Article97.chrysene_signed_expansion
#print axioms Article97.tetracene_signed_expansion
#print axioms Article97.chrysene_edge_count
#print axioms Article97.tetracene_edge_count
#print axioms Article97.chrysene_adjacency_det_value
#print axioms Article97.tetracene_adjacency_det_value
#print axioms Article97.zero_slack
#print axioms Article97.registry_count
#print axioms Article97.species_count
#print axioms Article97.support_count
#print axioms Article97.slack_registry
#print axioms Article97.strain_three
#print axioms Article97.strain_two
#print axioms Article97.strain_difference
#print axioms Article97.six_neighbors
#print axioms Article97.plaquette_xor
#print axioms Article97.box_faces
#print axioms Article97.feasibility
#print axioms Article97.interior_bits
#print axioms Article97.boundary_bits
#print axioms Article97.capacity_example
#print axioms Article97.radius_bound_constant
#print axioms Article97.capacity_slope_constant
#print axioms Article97.capacity_margin_constant
```